

TRANSPARENTLY EMBEDDING NON-COMPLIANT DATA IN A DATA STREAM

Background

[0001] High value digital content, such as multimedia, audio, and video content must be encrypted to prevent illicit access and usage of the data stream while it is transmitted between devices. Data streams are often transmitted in a lossy environment where data bits may be lost during transmission. Synchronization points in the data stream permit recovery from data loss and processing of the rest of the file. If the data stream is encrypted, then a playback system needs some way to synchronize key information used to decrypt the data stream. However, some standards, such as the Moving Pictures Experts Group (MPEG) standards, (See ISO/IEC 13818-1) provide no way to synchronize key information. Any attempt to embed key information into the data stream causes a standard-compliant decoder to fail. Alternatively, maintaining a parallel data stream for key information is complex and unwieldy. If the data stream is encrypted, there is no way to recover once data is lost. On the other hand, if the data stream is not protected with encryption, there is a way to recover. One should not have to choose between encrypting a data stream with no way to recover from data loss or not encrypting.

Brief Description of the Drawings

[0002] Figure 1 shows a block diagram of a system embodiment of the present invention for transparently embedding non-compliant data in a data stream.

Figure 2 shows a block diagram of an alternate system embodiment to Figure 1.

Figure 3 shows a block diagram of an alternate system embodiment to Figures 1 and 2.

Figure 4 shows a flow chart of a method embodiment of the present invention for transparently embedding non-compliant data in a data stream.

Figure 5 shows a flow chart of an alternate method embodiment to Figure 4.

Figure 6 shows a flow chart of an alternate method embodiment to Figures 4 and 5.

Figure 7 shows a block diagram of various data structure embodiments of the present invention within data streams, including transparently embedded non-compliant data.

Figure 8 shows a block diagram of various data structure embodiments of the present invention, including transparently embedded non-compliant data.

Detailed Description

[0003] Systems, methods, and data structures for transparently embedding non-compliant data in a data stream are described. In the following detailed description, reference is made to the accompanying drawings, which are part of this application. These drawings illustrate specific embodiments for practicing the present invention and reference numbers refer to substantially similar components throughout the drawings. The embodiments are described in sufficient detail to enable those skilled in the art to practice the present invention. Other embodiments may be used and structural, logical, electrical, and other changes may be made without departing from the scope of the present invention.

[0004] Figure 1 shows a block diagram of a system embodiment of the present invention for transparently embedding non-compliant data in a data stream 100. The system 100 comprises an authoring device 102 and a consumption device 104. The authoring device 102 is any combination of hardware and/or software capable of authoring content and transmitting a data stream. Some examples of authoring devices 102 are Digital Versatile Disc (DVD) authoring systems, Compact Disc (CD) authoring systems, satellite and television broadcasting systems, streaming web video systems, and the like. The consumption device 104 is any combination of hardware and/or software capable of receiving a data stream and consuming content. Some examples of consumption devices 104 are DVD players, CD players, televisions, video cassette recorders (VCRs), Personal Computers (PCs), handheld devices, and the like.

[0005] In one embodiment, the authoring device 102 uses key information to encrypt a portion of a data stream and the consumption device 104 uses the key information to decrypt the portion of the data stream. Key information is any information useful to any cryptographic algorithm, such as one or more random

numbers. The purpose of the key information is to make the encryption cipher non-repeatable. In another embodiment, the key information also includes other information such as navigation information, proprietary information, copy control information, copyright information, the identity of the source of the data stream, the client's account number, etc. In another embodiment, the key information changes on every frame or picture for video content. In this way, when it is played back, fast forwarded, or a random jump is made, the key information for each frame or picture is available. One example of a cryptographic algorithm is the Data Encryption Standard (DES), ANSI Data Encryption Algorithm (DEA), ANSI X3.92-1981 (R1998). Other examples are Rivest, Shamir, and Adleman (RSA) (See R.L. Rivest, A. Shamir, and L.M. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Communications of the ACM, v. 21, n. 2, Feb. 1978, pp. 120-126, Digital Signature Algorithm (DSA) Proposed Federal Information Processing Standard for Digital Signature Standard (DSS)," Federal Register, v. 56, n. 169, 30 Aug. 1991, pp. 42980-42982, and proprietary ciphers. In another embodiment, the system 100 further comprises a decoding device 106 in communication with the consumption device to decode the portion of the data stream. The decoding device 106 is any device capable of interpreting the data stream, such as an MPEG-compliant decoder.

[0006] In some embodiments, the decoding device 106 and the consumption device 104 are part of one system or device. In another embodiment, the consumption device 104 is capable of retrieving the key information from the portion of the data stream. In one embodiment, the authoring device 102 is a computer having a tool to create original data streams. The tool goes through the data stream and adds extra stuffing bytes to make room for key information, such as random numbers. The payload is encrypted and key information is put in on top of where the stuffing bytes were. Then, the content is transmitted to the consumption device 104. On the consumption device 104, generally, the reverse process is done. The data is processed, the key information is retrieved, the data stream is decrypted, the key information is replaced with stuffing bytes, and then the decrypted data stream is sent to a decoder. The decoder receives the packetized element stream (PES) header, the stuffing bytes, and the decrypted PES payload and then decodes as usual.

[0007] Figure 2 shows a block diagram of an alternate system embodiment to Figure

1. One system embodiment 200 comprises an authoring device 202, an encryption tool 204, and a consumption device 206. The authoring device 202 creates a data stream. The encryption tool 204 embeds key information near each synchronization point in the data stream and encrypts a portion of the data stream associated with each synchronization point. The consumption device 206 retrieves key information near each synchronization point in the data stream and replaces the key information with compliant data before the data stream is decrypted. In another embodiment, the system 200 further comprises a decoding device 208 to decode the data stream. In another embodiment shown in Figure 3, the system 300 further comprises a decryption tool 302 to use the key information to decrypt the portion.

[0008] Figure 4 shows a flow chart of a method embodiment of the present invention for transparently embedding non-compliant data in a data stream 400. Non-compliant data is data which does not strictly comply with a standard, such as one of the MPEG standards. The non-compliant data is, for example, navigation information or any other kind of information not strictly allowed by the standard. A portion of a data stream is received 402. The portion of the data stream is parsed to find a synchronization point within the data stream 404. An example of a synchronization point is a PES header. Non-compliant data near the synchronization point is retrieved 406. The portion of the data stream is decrypted 408. In another embodiment, the non-compliant data is key information that is used in decrypting. In another embodiment, the method 400 further comprises replacing the non-compliant data near the synchronization point with compliant data and decoding the portion of the data stream.

[0009] In one embodiment, the synchronization points comprise the PES headers. Every time the content has a new picture, there is a PES header, which is easy to find. Even if some data is lost, a receiving computer or a displaying computer realizes there is lost data or corrupted data and looks for the next PES header. Once the next PES header is found, everything is discarded up until that next PES header. That picture and subsequent data is decoded. The key information that was used to encrypt the content is in the PES header so that when the computer discarded data up until the next PES header, the key information needed to decrypt the current picture

is in the current PES header.

[0010] Figure 5 shows a flow chart of an alternate method embodiment 500 to Figure 4. The method 500 comprises parsing a data stream to find a predefined synchronization point within the data stream 502 and placing non-compliant data near the synchronization point in the data stream 504. The data stream is decodable by a compliant decoder 508, after the non-compliant data is replaced with compliant data 506. An example of a predefined synchronization point is a header. In another embodiment, the method 500 further comprises encrypting a portion of the data stream and transmitting the portion of the data stream. In another embodiment, the method 500 further comprises decrypting the portion of the data stream. In another embodiment, the non-compliant data is key information that is used in encrypting and decrypting.

[0011] In one embodiment, a data stream is stored as a data file on a computer. Software executing on the computer parses through the data file to find all of the synchronization points or PES headers and evaluates each one to determine if there are sufficient stuffing bytes to hold key information or any other information desired to be embedded. If there are sufficient stuffing bytes, then nothing is done and the next synchronization point is processed. If there are not enough stuffing bytes then some are inserted between the PES header and the PES payload following the PES header. After processing the data file and inserting stuffing bytes, if needed, the new data is written out to a new file. Then the new file is encrypted. An encryption tool follows a similar process, searching to all the PES header synchronization points, making sure enough stuffing bytes are there, embedding key information, and then encrypting the payload for all the synchronization points in the new file. At this point, the encrypted file is transmitted from the authoring device to another device. Any method of transmission is acceptable including transmission over an Ethernet connection, over a satellite, over standard television airwave broadcasts, and the like. After transmission, the consumption device performs a similar process in reverse. First, the data is parsed looking for PES header synchronization points. When one is found, data is sent to a decryption component which retrieves key information and uses the key information to decrypt the payload associated with the PES header synchronization point. The decryption component replaces the key

material with stuffing bytes as defined in the MPEG standards and the payload is, for example, decoded to make audio or video rendered for consumption.

[0012] Figure 6 shows a flow chart of an alternate method embodiment 600 to Figures 4 and 5. In one embodiment, a machine-accessible medium has associated content capable of directing the machine to perform the method 600. A first data stream is parsed to find a PES header 602. The PES header is associated with at least some payload data. The first data stream is copied to a second data stream 604. Compliant data is selectively inserted into the second data stream after the PES header to hold key information associated with the PES header 606. In another embodiment, the method 600 further comprises storing the first data stream and storing the second data stream. In another embodiment, the method 600 further comprises parsing the second data stream to find each PES header, embedding key information into each portion of the second data stream after each PES header, and encrypting each portion of the second data stream. In another embodiment, the method further comprises transmitting each portion of the second data stream. In another embodiment, the method further comprises retrieving key information from a portion of the second data stream, decrypting the portion of the second data stream with the key information, and replacing the key information with compliant data in the portion of the second data stream. In another embodiment, the method 600 further comprises decoding the portion.

[0013] Figure 7 shows a block diagram of various data structure embodiments of the present invention within data streams, including transparently embedded non-compliant data. To illustrate a sample data stream, a PES data stream is shown in Figure 7, but embodiments include many different types of data streams. The original data stream 700 comprises a PES header and a PES payload. The original data stream 700 is capable of being processed by any MPEG-compliant decoder. The data stream with inserted compliant bytes 702 has compliant data inserted into the PES header. The data stream 702 is capable of being processed by any MPEG-compliant decoder. The encrypted data stream 704 has an encrypted PES payload. The stuffing bytes have been replaced with key information needed to synchronize decryption keys with the data stream. Because the PES payload is encrypted, it is not capable of being processed by any MPEG-compliant decoder. However, the

PES header is decodable, if necessary. The data stream after decryption 706 has a decrypted PES payload. The key material has been used to perform decryption and then it was replaced with the original stuffing bytes, which are predefined in the MPEG standards. The data stream 706 is capable of being processed by any MPEG-compliant decoder. In one embodiment, the compliant data is defined as padding bytes either to make a data packet a certain size, to help alignment, or the like.

[0015] In some embodiments, data streams 700, 702, 704, and 706 are used to control access to multimedia video and audio data streams. By utilizing the stuffing bytes as defined in the MPEG standards, key information and navigation information is embedded without having to modify other components in a playback system that are not specifically involved in the encryption, decryption, and navigation processes. In another embodiment, there are stages that content goes through on a content preparation computer. The content in a format similar to the encrypted data stream 704 goes out over some transmission protocol, is received by a consumption device or playback computer. Then, the playback component and the decryption tool retrieve the key information and use it to re-initialize a cipher. The stuffing bytes are written back in and the payload is decrypted. Then, it is fed to the decoder which processes the data stream, ignoring all the stuffing bytes. In this way, the key information travels with the data stream from content preparation to content consumption, avoiding having to construct any kind of back door way of getting the information into the application.

[0016] Figure 8 shows a block diagram of various data structure embodiments of the present invention, including transparently embedded non-compliant data. In one embodiment, a data structure 800 comprises a header, key information associated with the header for use in decryption, and a payload associated with the header, the payload capable of being encrypted using the key information. In another embodiment 800, the key information in the header replace compliant data, after encryption. In another embodiment 802, compliant data replaces the key information associated with the header, before decryption. In another embodiment, 804, the header, compliant data, and decrypted payload are capable of being decoded by a compliant decoder. In another embodiment 806, the header is a PES header and the payload is a PES payload. In another embodiment, a data stream (such as the

data streams 700, 702, 704, and 706 shown in Figure 7) is stored on a machine-readable medium and comprises at least one data structure 800, 802, 804, or 806.

[0017] In another embodiment, spare bytes, stuffing bytes, and/or additional bytes as they are defined in the MPEG standards, 1994 are used for the PES headers to store key information. In this embodiment, a transmitting computer is capable of modifying the original data stream. The transmitting computer embeds the necessary additional information in each PES header that needs to be used as a key information synchronization point. The file is encrypted and the key information is embedded in the stuffing bytes. At this time the contents of the stuffing bytes do not conform to the MPEG standards, but that does not matter because only the decryptor on the receiving computer is capable of processing the data stream while the data is encrypted. The PES header, which contains the stuffing bytes is not encrypted, only the payload following the PES header is encrypted. During decryption on the receiving computer, the key information is retrieved from the stuffing bytes and the payload data is decrypted. Then, the stuffing bytes are modified to replace the key information with the appropriate data values to conform to the MPEG standards so that a playback component is capable of correctly processing the decrypted data stream. As a result, there is no need for a parallel data stream because the key information is stored directly in the MPEG stuffing bytes.

[0018] It is to be understood that the above description it is intended to be illustrative, and not restrictive. Many other embodiments are possible and some will be apparent to those skilled in the art, upon reviewing the above description. For example, other embodiments include any protocol or standard for transmitting data streams that defines synchronization points in a data stream, MPEG-1, MPEG-2, MPEG-4, future MPEG standards, various MPEG applications such as video phones, digital television broadcasting, compact-disc video (CD-Video), digital video disc (DVD), personal video recorders (PVR), networks, IEEE 1394-1995 interfaces, broadband networks, asynchronous transfer mode (ATM), and more. Other embodiments use spare, stuffing, padding, and other non-essential bytes for replacement. Therefore, the spirit and scope of the appended claims should not be limited to the above description. The scope of the invention should be determined

[illegible]